(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 049 387**
A2

(12) # EUROPEAN PATENT APPLICATION

(21) Application number: 81107289.1

(22) Date of filing: 16.09.81

(51) Int. Cl.³: G 06 F 13/00

(30) Priority: 06.10.80 US 194639

(43) Date of publication of application:
14.04.82 Bulletin 82/15

(84) Designated Contracting States:
DE FR GB IT

(71) Applicant: International Business Machines Corporation
Armonk, N.Y. 10504(US)

(72) Inventor: Evans, Charles Wesley
10604 Glass Mt. Terrace
Austin Texas 78750(US)

(72) Inventor: Flusche, Frederick Otto
Box 156 Matuk Drive
Hyde Park New York 12538(US)

(72) Inventor: Messina, Benedicto Umberto
40 Round Hill Road
Poughkeepsie New York 12603(US)

(72) Inventor: Richardson, Ethel Louise
11 Hammersley Avenue
Poughkeepsie New York 12601(US)

(72) Inventor: Robinson, James Robert
Hickory Hill Road
Clinton Corners New York 12514(US)

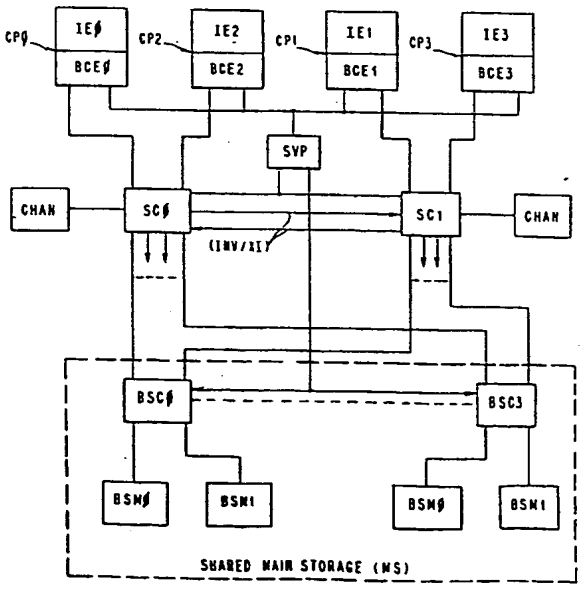
(72) Inventor: Wetzel, Joseph Albert
333 Rt. 32 South
New Paltz New York 12561(US)

(74) Representative: Hawkins, Anthony George Frederick
IBM United Kingdom Patent Operations Hursley Park
Winchester Hants. S021 2JN(GB)

(54) **Multiprocessor system with cache.**

(57) A data processing system includes a plurality of central processing units (CP0 to CP3) each including an instruction execution unit (IE) and a buffer control unit (BCE). Each buffer control unit includes a store in cache, a cache directory and cache controls. The processing units are coupled to a shared main storage system (MS) through system control means (SC0, SC1). The system control means includes a plurality of copy directories, each corresponding to an associated one of the cache directories. When a processing unit issues a clear storage command, the address thereof is directed to all of the copy directories. For each one found to refer to the address, an invalidate signal is sent to the associated cache directory to invalidate its corresponding entry, if permitted by its processing unit, and a resend signal is sent to the processor which issued the clear storage command. The entries in the copy directories corresponding to invalidated entries in the cache directories are then also invalidated. The procedure continues until all corresponding entries in the directories are invalidated, at which time the directory search results in an accept signal which enables the main storage area defined by the command to be cleared.

FIG. 1



EP 0 049 387 A2

0049387
TITLE MODIFIED
see front page

DATA PROCESSING SYSTEM

The present invention relates to data processing systems, and in particular to arrangements for clearing main storage areas therein.

The clearing of main storage using a move long (MVCL) instruction has been in public use for many years in IBM System/370 data processing systems using store through (ST) caches.

Recently the store-in-cache (SIC) has been found to provide better system performance than store through (ST) caches in multiprocessor (MP) systems because the SIC has less bandwidth requirements to main storage than the ST. A processor having a SIC does not see main storage when it performs a write operation since the processor writes into the SIC, unlike a ST cache processor which writes directly in main storage. Thus, a processor with a SIC can only indirectly cause writes into main storage when cache misses occur, which are only after the addressed entries in the cache are filled with changed data, or when castout is required by an I/O or another CPU request.

Multiprocessors (MPs) have cache problems (independent of the type of cache used) occurring when clearing MS shared by plural processors. For example, it is well known that if any processor in a MP clears data in MS, invalidation must be performed on any data in any processor cache in the MP having an address in the cleared MS area. However, it is not well known that the MP may be made inoperable in certain circumstances if any of its processors is permitted to clear an area in MS and make an untimely cache invalidation of data for the cleared area. The reason for this is that a subtle MP architectural rule may be violated, which is that all MS sharing processors must be able to access the same version

of MS data. This rule assures that any MP will obtain the same results with the same data. For example, a late cache invalidation may permit a processor to be using uncleared data in its cache when the corres ponding data has already been cleared to a different value in MS by another processor (allowing two processors to be operating at the same time on different data from the same MS address, whereby the MP data result of such operation becomes dependent on the time of invali dation which is transparent to the system users), and therefore MP data results become unpredictable.

Early cache invalidation is as bad as late invali dation and likewise may cause unpredictable results. For example, if one processor is to clear an MS area and its first act is to invalidate all processor caches in the MP, any other processor may still be able to fetch a part of the yet uncleared MS area to replace its invalidated data with the yet uncleared data; and such data in the other processor cache would thereafter not be invalidated, so that the other processor is operating with different data for the MS address which had shortly thereafter been changed in MS by the clearing processor without being again invalidated in the other processor cache.

Accordingly, the changing of an MS area must be synchronised in some adequate fashion with the invalidation of any data for that MS area in all processor SIC caches. This problem is solved in the currently used ST caches in the IBM System/370 M168 or 3033 MPs by the act of storing through the cache directly to MS for each doubleword changed by any processor and invalidating the same line in the cache of the other processor in the MP before the other processor can store its next doubleword into MS.

However, when the processors in an MP use a SIC rather than a ST cache, this invalidation synchronization problem (which is not readily apparent) becomes more difficult, because processor changes to MS data are not directly made in MS but are only made in a processor's SIC. Changed cache data eventually is stored in MS by a SIC whenever the SIC becomes filled and its LRU (least recently used) circuit forces some cache data to be stored into MS to make room for new data in the cache. The result is that the invalidation of correspondingly addressed data in another processor's cache need not be done until some processor or I/O channel provides a store request to the data. Before the store may take place in the requested cache, a broadcast or cross-interrogation must be made to all other caches to search all other caches for the address of the data; and if data at that address is found to be changed in any other cache, it is first put back into MS and fetched into the requested SIC (or directly transferred to the requesting SIC when being put into MS) so that the requesting cache will have the latest version of the data at that address before it performs the processor requested store operation. Therefore only one processor in the MP is permitted to have data which is in the process of being changed; and such data is held exclusively in the cache in which it is to be changed. Nevertheless, plural processors in the MP can be asynchronously fetching the same data in their different caches for read only operations without any cross interrogation or invalidation, as long as they do not attempt to change the data. Therefore, as long as the data is held read-only in the MP caches, no invalidation is required.

Castout controls for an MP system having copy directories and command queues in its system controllers is disclosed and claimed in U.S. Patent Specification No. 4,136,386.

The exclusive/read-only state of data in a SIC and cross-interrogation between processor cache directories in an MP are disclosed in U.S. Patent Specification Nos. 3,735,360, 3,771,137 3,723,976, and 4,056,844. These all deal with different inter-processor controls for handling data to be stored in a SIC in an MP. These patents do not deal with the problems existing when a processor having a SIC attempts to clear part or all of MS.

This invention is based on the recognition that a processor with a store in cache (SIC) performs poorly when used for clearing storage and provides a solution to this problem A SIC must first fetch data from MS before its processor can store data, for reasons previously discussed requiring the SIC to have the latest version of MS data. Thus, with normal SIC controls, a MS clear operation by a processor (such as by execution of a MVCL instruction) would require the data in the MS area being cleared by a processor first to be fetched into the SIC, cleared in SIC, and only eventually stored back into MS when (1) the space in SIC occupied by the cleared data is required by the LRU circuit for newly fetched data from MS, or (2) in a MP when changed data at the requested MS address is requested by I/O or another processor.

It is an object of this invention to improve processor efficiency in executing a clear storage (CS) command by preventing any processor SIC from casting out data which would be stored in an area of MS which is to be cleared by the CS command.

According to the invention, there is provided a data processing system including a plurality of central processing units, each having a store in cache, a cache directory and store in cache control means, and including system control means coupled between the processing units and a shared main storage system, character-

ised in that the system control means includes a plurality of copy directories, each being a copy of an associated one of the cache directories, first means for simultaneously searching all of the copy directories for an address contained in a clear storage command issued by a processing unit, second means coupled to the copy directories for generating invalidation signals for each copy directory containing the searched address and coupled to the respective cache directories for invalidating the corresponding address entries therein and third means coupled to the copy directories to provide a resend signal to the processing unit issuing said command when any of the copy directories contain the searched address or an accept signal thereto when none contain said address, and in that each said store in cache control means includes means responsive to said invalidation signals to invalidate the cache directory address defined by said command and means responsive to said invalidation to invalidate that address in the corresponding copy directory, said processor issuing said command being responsive to a resend signal to re-issue the command to initiate a copy directory search or to an accept signal to clear the main storage area defined by the command.

In an embodiment to be described hereinafter, when any processor in an MP executes a clear storage (CS) command, controls are provided to synchronize directory search operations in all processors. The synchronization is initiated by a cross-interrogation (i.e. broadcast) signal to directories for all processor SICs in the MP from any processor which issues a CS command. To reduce cache interference due to cross-iterrogation, a copy of each processor cache directory is maintained by the storage control for purposes of cross-interrogation (XI), so that all XI requests are made to and processed by the copy directories rather than by the processor cache directories. Then the only time that any processor cache directory is interfered with for purposes of cross-interrogation is when its copy directory finds that the processor directory contains data to be invalidated, i.e. finds a XI conflict

exists. Only a small percentage of XI requests among copy directories will find XI conflicts, so that the processor cache directories are not bothered by most XI requests.

The CS command from a processor is inhibited from going to the processor cache directory, as would normally occur for other processor storage requests. Therefore, the CS command execution is not delayed when the processor cache directory is in use by a prior request. Also, the CS command need not immediately invalidate any conflicting line in the issuing processor cache directory before the directories for other processors can be searched. But the processor cache controls immediately transfer the CS command to controls for its copy directory, which then sends a request to MS to clear the data from MS, and sends a XI request to all other copy directories in the MP system to initiate a search of all copy directories in order to determine if any XI conflict exists. If any conflict is found in any copy directory in the MP, the processor which issued the CS command is signalled to reissue the CS command in order to properly time the execution of the CS command in relation to all conflicting processors.

If a conflict is found in any one or more of the copy directories, in the MP, the controls for each conflicting copy directory send an invalidate request to its respective processor cache directory to request the invalidation of the conflicting data in the processor directory. Then the controls for the copy directory for the issuing processor sends a resend command to its respective processor cache directory. The invalidation request to any processor cache is accepted when its cache priority circuit determines the processor cache directory is available to perform the invalidation request, i.e. it is non-busy and the invalidation request is its highest priority outstanding request. For example, a processor

cache may be busy awaiting the translation of a prior processor request and not be able to immediately accept an invalidation request.

If any conflict is signalled to the copy directory controls for the issuing processor, the controls also send a resend command to the processor instruction execution (IE) unit to prevent the processor from issuing its next command by causing the processor to reissue the CS command, which starts a next iteration of an invalidation looping operation, in which the CS command is re-executed in the same manner and again cause a cross-interrogation to and search of all copy directories. As long as any copy directory in the MP finds a conflict during an iteration, it sends a conflict signal to all copy directory controls, and the conflict signal is used by the issuing copy directory controls to again issue the resend signal to its IE unit, which again reissues the CS command. In this manner the looping operation continues the execution of the CS command in the issuing processor. During each loop, an invalidate request is generated from each conflicting copy directory to its processor copy directory, i.e. having a cache with data to be cleared which has not yet been invalidated. The loop iteration in which the last cache in the MP has invalidated its conflicting data is followed by the last iteration of the loop in the issuing processor, which recognizes that no conflict then exists in the MP; and its copy directory controls send an accept signal to the issuing IE unit (instead ot a resend signal), which then stops reissuing the CS command to end the looping, and no MS cancel signal is sent to MS. The clearing operation then takes place in MS, and the execution of the CS command is completed for all caches in the MP.

A copy directory invalidation request need not be accepted by a processor's cache directory, in which case the request does not

cause an invalidation of the conflicting data in its processor cache directory. That is, a processor's cache priority circuit may not accept its copy directory's invalidation request during an iteration of the loop, which continues the MP conflict into the next loop iteration when it will again be detected by the synchronized directory search, and the issuing processor will reissue the resend command to cause the looping to continue until no conflict exists in the MP. The continuation of the invalidation looping is dependent upon not invalidating the conflicting data in the copy directory until after the invalidation is done in its processor cache directory. Therefore, the conflict remains detectable in the copy directory during the search in the next loop iteration, which is essential for continuing the loop iterations as long as the data is not invalidated in the processor cache directory. In this manner, all conflicting processors are synchronized by the looping so that no processor which has conflicting data is permitted to store the data in the MS area to be cleared.

The conventional unit of data transferred between the cache and MS is called a line (or block), which comprises a fixed number of contiguous bits in MS which are addressed as a unit in the cache directory. A line may be comprised of N transfer units, in which a transfer unit is one parallel transfer of data on the bus between MS and the cache. For example, a line may comprise sixteen double words (DWs) which may be transferred a DW at a time in sixteen sequential transfers to the cache on a MS bus which is one DW wide.

When an area in MS is to be cleared under program control (e.g. MVCL instruction), the area may be less than, equal to, or greater than a cache line. The number of lines in the area to be

cleared is determined by the instruction execution (IE) unit in a processor executing a clear storage instruction. In the preferred embodiment of this invention, the IE generates a special command called a clear line (CL) command, which is a specific form of the more general CS command previously discussed herein. The IE will issue one CL command for each line to be cleared in MS. The procedure used for the CL command is as previously described for the CS command. Thus, the CL command is provided by the IE to the processor cache directory controls (BCE) with the address of the line to be cleared. The BCE inhibits a search of the processor cache directory but sends the CL command to the associated copy directory controls. The receiving copy directory controls generate and send a XI search request to every copy directory in an MP system. A synchronized search results in all copy directories in the MP, even though the processors in the MP are asynchronous with each other. The XI synchronization is obtained by respective priority circuits in the respective copy directory controls which give high priority to an XI request. Because of the synchronization, if a conflicting line address is found in any copy directory, its conflict signal is provided to the issuing processor during the synchronized search, in order to continue execution of the CL command in the issuing processor in relation to every conflicting processor in the MP, and to prevent the clearing of the line in MS until the address of the line to be cleared is invalidated in every cache in the MP.

Any non-issuing processor performing the synchronized search without finding the conflicting line in its cache is thereafter free to use its cache. But if any processor without a conflict should make a fetch of the conflicting line before completion of the execution of the CL command by the issuing processor (and therefore before the line is cleared in MS), the newly conflicting

line will be detected by the search during the next loop of the CL command and will be invalidated before the concurrent CL command completes its execution.

In this manner, the cache invalidation procedure occurs in a way which prevents architectural violation of the consistent results rule in an MP. That is, all processors are inhibited from effectively accessing a line being cleared in MS when accessing their SICs during the execution of a CL command. The copy directory search synchronization and the looping reissuance of the XI requests assure all processors in the MP that none of them can cause any line invalidation in their SIC either too late or too early in relation to the time the line is actually cleared in MS, so that all processors are assured that their program execution results will not be made inconsistent by improper timing among the line invalidation in the different caches and the clearing of the line in MS. The CL cache search synchronizations effectively prevent the asynchronous processors from processing a line being cleared in MS until all processors in the MP invalidate the conflicting line from their caches. If any processor has a concurrent request for a copy directory search, the CL XI search for any conflicting line will obtain priority over and thereby preceed the copy directory search for the concurrent processor request in order to obtain substantial synchronization for all XI directory searches.

The invention will now be described, by way of example with reference to the accompanying drawings, in which:-

FIGURE 1 is a block diagram of a multiprocessing system;

FIGURES 2A, 2B, 2C and 2D illustrate the types of information found in various commands used in the described embodiment of the invention;

FIGURE 3 illustrates diagramatically the overlapping storage hierarchies found in a multiprocessing (MP) system having processor exclusive caches and a shared main storage (MS);

FIGURE 4 illustrates controls found in the instruction execution (IE) of any one of the central processors (CPs) shown in FIGURE 1 of which CP0 is used as the exemplary processor;

FIGURE 5 illustrates the processor cache controls (BCE) found in each central processor;

FIGURES 6A and 6B illustrate the pertinent controls found in each system controller (SC) shown in FIGURE 1;

FIGURE 7 illustrates in block form the data path found in the system controller and a BSM controller (BSC) in the data processing system;

FIGURE 8 shows in detail the cache cast-out and invalidate controls 64 represented as a block in FIGURE 5;

FIGURES 9A, 9B, 9C, 9D and 9E represent timing diagrams showing the sequencing of operations in the described embodiment; and

FIGURE 10 shows another BSM controller (BSC) arrangement which provides a faster clear line operation than in FIGURE 7.

The multiprocessor (MP) illustrated in FIGURE 1 comprises four central processors CP0 through CP3, in which each CP includes an instruction execution (IE) unit and a buffer control unit (BCE). The IE includes the hardware which issues a clear storage (CS)

command which in this particular embodiment is more specifically called a clear line (CL) command because it controls the clearing of one line in MS. The IE issues the CL command to its BCE. The BCE includes a processor store-in cache (SIC) with its associated processor cache directory and associated processor cache controls which are exclusively used by the associated CP.

This invention solves an MP integrity problem in the clearing of shared main storage by requiring that the unit of invalidation in the caches must not be less than the unit of area being cleared concurrently in shared main storage. In this embodiment, the cache line is the unit which is cleared in MS in any single operation and the unit which is invalidated in any cache in which a conflict occurs.

Two system controllers SC0 and SC1 each connect to two CPs and to the shared main storage (MS), so that either connected processor can access any shared area in main storage. Each SC also connects I/O to the shared main storage.

The shared main storage comprises of a plurality of BSM controllers BSC0 through BSC3 in which each BSC connects to two BSMs 0 and 1. The four BSCs 0-3 are each connected to each SC.

An invalidate cross-interrogate (INV/XI) bus connects from each SC to the other SC. The INV/XI bus communicates all cross-interrogation (XI) requests and all castout (CO) and invalidate (INV) signals between the SCs in order to coordinate the required castouts and invalidations in the directories.

Each SC contains a pair of copy directories. Each copy directory contains an image of the contents of a corresponding processor cache directory in one of the BCE's connected to the respective SC.

FIGURE 3 illustrates the storage hierarchies I through IV found between the respective CPs and the shared main storage. Thus, each CP interfaces the apex of a hierarchy comprising the BCE cache found with the respective CP. Each BCE interfaces its corresponding copy directory. The base of the hierarchy is the shared MS with which each CP communicates. The data transfer between each CP and the shared MS is performed a line at a time, in which each request from the BCE to the shared main storage is for a line of data or instructions in MS when a CPU request is not available in the BCE cache or the BCE cache does not have the latest changed version of the requested data.

The copy directories handle all cross-interrogation (XI) requests so that the processor cache directories in the BCEs will not need to handle the XI interrogation requests and can thereby better service their respective CPs. CO/INV busses communicate both invalidate and cast-out requests from other CPs so that conflicts between the CPs may be avoided when accessing shared main storage.

FIGURE 4 illustrates the instruction execution (IE) unit in CP0 in regard to the issuance of a clear line (CL) command by CP0. Each of the other CPs 1-3 has a IE identical the the IE0. The described IE unit is basically similar to IE units in microprogrammed processors having a microinstruction control store containing microprograms. A unique microinstruction (designated the clear line microword) is provided in control store 13 for initiating the clearing of a single line in the shared main storage. The

clear line (CL) microword is issued via the microinstruction select and decode unit 14.

Two different ways of controlling the issuance of the CL microword are represented in FIGURE 4. One way accommodates the MS validation problem previously discussed herein. The other way accommodates normal program usage of the CL command for clearing a designated area in MS. A signal from a system service processor (SVP), usually associated with a system console, is provided on lines 5 in FIGURE 4 to microinstruction select and decode unit 14 to cause unit 14 to address and initiate execution of a special SVP CL micro-instruction sequence in control store 13. No access to MS is involved in the initiation and execution of this special SVP CL microprogram in CP0. This micro-program issues and executes a CL command issues and executes a CL command for every line address in MS in order to validate all of MS without any fetching of data from MS, and without any access to the SIC in CP0, which is inhibited by a gate 59 in FIGURE 5. The special SVP CL microprogram is shown in control store 13 beginning with the SVP CL microword and ending with the SVP END OP microword. A microword in this microprogram loads a pad byte into the IE data register 17, and other microwords increment the line address in operand address register 16 to every line address in MS.

On the other hand, the invention also provides for the CL command to be issued under normal program instruction control by any instruction which can clear or otherwise propagate a pad character over an area in MS. A second CL microprogram is represented in control store 13 for use by instructions in programs being executed by CP0. The second CL microprogram is shown in control store 13 beginning with the INST CL microword and ending with the INST END OP microword. The INST CL microword and the SVP CL microword may be identical except for the control store address they

define for selecting the next microword in their respective micro-program. An example of a normal program instruction which may issue the CL microword is the move long (MVCL) instruction which is found in the instruction set for the IBM System/370 model machines. The MVCL instruction involves the use of four general registers (GRs) which are found in a local store 12 and which are selected under GR select addressing control 19. The operation of the move long instruction is described in the IBM System/370 Principles of Operation (Form Number GA22-7000). The MVCL instruction can propagate a pad byte in its source operand into its sink operand, and the sink operand may have a variable length up to 16,777,215 bytes. The source operand address is found in general register R2, and the length of the source operand is found in general register R2+1 bit positions 8-31. When the source operand address in general register R2+1 is zero, the pad byte found in bit positions 0-7 of register R2+1 is moved into the sink operand in each byte position up to the end of the source length defined in register R1+1. General registers R1 and R2 are even-numbered registers and R1+1 and R2+1 are odd-numbered registers in local store 12. Exemplary hardware shown in FIGURE 4 determines the number of lines of the pad character to be stored in MS during the execution of a MVCL instruction, the MS address of each line for which the IE unit issues a clear line command, and the number of lines remaining to be moved to the sink area in order to keep track of how many consecutive clear line commands should be issued.

The line size is set by a microword that stores a line size constant into the content of line size constant register 35, which for example may be 128 bytes representing 16 double words. A current line address register 32 initially receives the R1 address from local store 12 via gate 31 under control of microinstruction

select and decoder unit 14, and provides the MS address in each CL command while executing the MVCL instruction. Register 32 is updated with the address for each next CL command by an adder 33 which receives the current line address in register 32, and adds to it the line size constant received from register 35 via a gate 36 each time a command accept trigger 42 receives a command accept signal from SC0. The updated output of adder 33 is loaded into register 32 to form the next CL command address. A line counter 38 maintains a count of the number of lines remaining in the execution of the MVCL instruction. The initial value in counter 38 is provided by a divider 37 which divides the sink area length gated from R1+1 by the line size constant from register 35 to determine the initial number of lines to be stored in the sink operand of the MVCL instruction. Line counter 38 is decremented by 1 from line 44 each time a CL command is executed, which is signalled by an output from a command accept trigger 42. When the output of the decrementing line counter reaches zero, it signals microinstruction select and decoder 14 to indicate when the execution of the MVCL instruction is completed.

The pad character in MVCL register R2+1 is provided from local store 12 into an IE data register 17 and is replicated throughout the byte positions of doubleword register 17 under control of decoder 14. Operand address register 16 receives the initial line address from register R1, and thereafter receives the line address from adder 33 so that register 16 reflects the main storage address of the current line which is the subject of the current CL command. A PSW key register 11 enables the CL command to determine that there is no protection violation when it accesses a line in MS.

Upon the completion or termination of each CL command issued by the IE, a successful action completed (SAC) signal is received

from its BCE. The SAC signal causes the IE to issue the next command, which may be a reissue of the same command when circumstances require. When command accept trigger 42 is set, the IE issues the next command, and when command resend trigger 41 is set, the IE reissues the same command. A CL command may be reissued several times before it is finally completed. The accept or resend signals are received from SC circuits in FIGURE 6B to set triggers 42 or 41, and to pass through an OR circuit 45 to condition an AND gate 46 which is then enabled by a timely SAC signal to permit register 16 either to again ingate the same MS line address, or to ingate the next line address, depending on gate 36 being activated by command accept trigger 42. Also a command trigger 47 is set by the SAC signal to also control the ingating into register 16. Each time a CL command is issued or reissued, its decoder unit 14 activates its command request line 25 to signal the BCE that another command is available for BCE execution.

The pertinent details in BCE0 are illustrated in FIGURE 5 and are representative of the other BCEs 1-3 in the MP system. BCE0 contains a processor store-in cache (SIC) 63 which is the high speed storage buffer for CP0. All CP0 storage requests are provided to BCE0 which determines whether the requested data is currently in SIC 63.

The IE0 is capable of providing all processor commands. Most of the issued commands are storage access commands, but there are also special access commands which operate differently from commands that normally access storage, such as the CL command. A normal storage access command will have its address immediately provided by the BCE to its SIC directory 62 to determine if requested data is available in SIC 63. However, when the BCE receives a CL command, it does not search its SIC directory 62, but instead

inhibits a search of SIC directory 62 by activating a CL address inhibit gate 59 by means of an output from a CL trigger 58 in command controls, which include a decoder 57 that receives all commands on line 26 from microinstruction select decoder 14 in FIGURE 4.

The CL command is one of up to 256 commands which may be decoded. When a command is decoded, a particular output from decoder 57 sets CL trigger 58. Whenever the IE outputs a command, it activates a line 25 if it is a special command request (such as for a CL command), or activates a line 27 if it is a normal storage request. Lines 25 and 27 are provided as request inputs to a BCE priority circuit 28 which determines which of contending requests will be accepted for execution by the BCE during the current machine cycle. Other request inputs (not shown) internal to the BCE are provided which are not pertinent to this invention.

The output of priority circuit 28 is also provided on line 29 to set a command request trigger 56 to indicate a request to a PIR priority circuit 129 in FIGURE 6A that a command will be available from register 55 in BCE0 during the next machine cycle. A similar command being formulated in the BCE2 of CP2 would set the CP2 identifier bit on in the command. Hence, the SC can use the CP ID field in the command to recognize which CP issued the command.

Although IE issues a clear line command, it is inhibited from reaching SIC directory 62 and it is provided to a command data (CMD/DATA) register 55 in the exemplary format illustrated in FIGURE 2A which component parts are gated into register 55 by the

set of gates which reference FIGURE 2A wherein the four gates respectively receive the four information components shown in FIGURE 2A. Thus, the first information component, PSW key, is gated on lines 21. The absolute address of the line to be cleared is outputted from a register 53 which receives the output from an address translation and prefixing controls 52 having the dynamic address translation (DAT) circuitry 52A and a dynamic look-aside address translation (DLAT) array 52B. Controls 52 receive the logical address provided by the IE to BCE address register 51 on lines 22 provided from operand address register 16 in FIGURE 4. The formulated command is gated out of register 55 on bus 69 to processor input register (PIR) 87 found in FIGURE 6A.

A BCE0 command is provided from bus 69 through gate 70 to PIR 87 when permitted by a PIR priority circuit 129 which determines which command will be received by PIR 87 when BCE0 and BCE2 are simultaneously providing commands. Circuit 129 may provide alternating priority, or it may be a circuit such as that shown in U.S. Patent Specification No. 4,152,764. If no contention exists, a command is immediately accepted into PIR 57. Then the command is provided into the cross-interrogate register (XIR) 91 from which it is used to search all copy directories in the MP system. For searching the local copy directories, it is transferred into a directory register 131. For searching remote copy directories, XIR 91 is simultaneously transferred on line 144 to the XI bus from which it is received by the remote SC and put into its directory register 131, when it gets priority from the XIR priority circuits 130 in all SCs.

Therefore, PIR 87 also has an input from BCE 2 provided to gate 71, in order to receive commands from CP2 under the control of priority circuit 129.

Whenever a command received into PIR 87 has a one bit in a CL bit position of a command, it will activate a gate 89 to pass the command from PIR 87 to a storage input register (SIR) 90, from which is put on a main storage bus 141 as a request to main storage.

Before any command can pass from either the local or remote XIR 91, it must obtain priority from an XIR priority circuit 130 in every SC. Circuit 130 determines the priority of contention for searches in all SC copy directories, of which the local copy directories are shown in FIGURE 6B as directories 116 and 117. Similar copy directories exist in each remote SC. The requesting inputs to circuits 130 are the inverted output from a local busy trigger 134, the local cross-interrogate (XI) signal from an operation decoder 88 when it determines the command operation code in PIR 87 requires a directory search conflict determination, and a remote SC busy signal, a remote XIR request signal from the INV/XI bus. Also the XIR priority circuit 130 in each SC synchronizes its searches with the XIR priority circuit in each other SC via signals between them on a XI search synchronization bus 130A.

The output of priority circuit 130 determines whether a local command in XIR 91 or a remote command from the remote XIR register is permitted to input into directory register 31 via gate 132 or 133, respectively. The priorities in circuit 130 and signals on bus 13A are such as to cause the same request to be searched simultaneously in all local and remote SC copy directories. The synchronization is aided by the priority selection in circuit 130 in which the local and remote SC busy lines inhibit a search for a respective command if the SC is busy during that machine cycle.

PRIORITIES OF SEARCHES IN LOCAL SC

Local Cmd	Local Busy	Remote Cmd	Remote Busy	Simultaneous Search In Local Directories
No	No	Yes	No	Search Remote Cmd
Yes	No	No	No	Search Local Cmd
Yes	No	Yes	No	Search SC0 then SC1 Cmds Alternately
No	X	No	X	Wait
X	Yes	X	X	Wait
X	X	X	Yes	Wait
X	No	X	No	Wait

X = Don't care if yes or no

If any command is selected, one of the two output lines from circuit 130 activates either gate 132 or 133 to select the local or remote XIR command address for searching in all copy directories. The two output lines are received by an OR circuit 130B, which provides an output that sets the local busy trigger 134 to indicate all copy directories in the system are busy. A timeout circuit 129 also is activated by the set signal; and after a time period equal to the search time for all directories, it outputs a reset signal to trigger 134, putting it in its non-busy state. The non-busy signal from trigger 134 on line 134A resets the directory conflict triggers 124 and 134 in FIGURE 6B.

The execution of a storage command also includes the key checking needed to assure that there is no main storage protection violation (SPV) by a CL command. This is done by providing a

remote SC command's address and key on XIR line 144 to a local SC storage protection array and controls 92 to select the protect key for the storage block being accessed and compare it to the PSW key in the current command. Likewise, a remote XIR command's address and key is provided on line 143 to controls 92 to protection check a remote request. The storage protect controls 92 determine if a storage protect violation (SPV) exists, and then provide a signal which sets a SPV trigger 96 that provides a SPV signal on line 75 to BCE0 and also to FIGURE 6B in the SC. A storage protect violation actuates release controls 93 to end the execution for the command causing the violation by setting a release trigger 94 and a status register 95.

When a CL command is accepted by priority circuit 130, it is transferred into directory register 131, from which it outputs the CL address on lines 131A and a CL command signal on line 131B to FIGURE 6B. The CL address is received by directory address register 114 which simultaneously searches both copy directories 116 and 117. Address decoder 114 decodes the class address in both directories and transmits the set selection bits to comparators 118 and 119 at the outputs of the directories to determine whether the requested address is in each directory in the manner conventional for set associative directories. If the CL address on line 131A is found in either directory, its comparator output sets an associated directory conflict trigger 124 and an invalidation request trigger 123 for CP0, or conflict trigger 134 and invalidation request trigger 135 for CP2. The setting of either conflict trigger 124 or 136 in the SC causes a conflict output signal on line 137A from OR circuit 137 which is transmitted to the remote SC from FIGURE 6A.

An MP conflict is detected by an OR circuit 138, which receives the outputs of both conflict triggers 124 and 136 in the local SC and also receives any conflict request on line 137B of the

XI bus from the remote SC which is provided by a remote corresponding OR circuit 137. Accordingly, an output from OR circuit 138 indicates whether any cache conflict exists in the multiprocessing system.

The MP conflict signal from OR circuit 138 is provided to a circuit 142 which generates a resend or accept signal which is sent to the CP which issued the current command. The CP ID field in XIR 91 signals which CP issued the current command in XIR 91. Circuit 142 includes a command resend trigger and a command accept trigger. The command resend trigger is set for one cycle if the CL command cannot be immediately completed. The circuit 142 is conditioned by the existence of a CL command signal which is provided to its AND gates 98, 99 and 106. The accept trigger 104 is set if the CL command is about to be completed which occurs when no MP conflict exists, which occurs when AND gate 106 is activated by the output of inverter 125 to provide a signal through OR circuit 103 that sets the command accept trigger. If a MP conflict does exist, AND gate 99 is activated by the output of OR circuit 138 and by the CL command signal as long as there is no storage protect violation (which is indicated by the output of inverter 97). If a storage protect violation does occur, then the command is immediately ended by the SPV signal on line 75 activating AND gate 98 which sets the accept trigger 104 to complete the execution of the CL command during an MP conflict when the storage protect violation exists.

When invalidation request trigger 123 is set, an invalidation command is formulated in a register 120 for CP0. Likewise, if SC copy directory 2 sets sets invalidation request trigger 135, an invalidation register 121 formulates an invalidation command for

CP2. The information in the invalidation command is illustrated in FIGURE 2C and it is transmitted from the SC to the respective BCE having a conflict in its SIC directory (castout fields in this SC to BCE command are not pertinent to this invention and are not described in detail). The content of register 120 therefore includes the directory class address obtained from the output of DAR 114, while that set address is obtained from the output of comparator 118, the clear line bit is set from line 131B, while the castout bit is obtained from the respective CPs C0 controls. The invalidate bit is not in register 120 but is provided on line 71 from the output of invalidation request trigger 123.

Invalidation register 121 has its command formulated in the same manner as in register 120, except that register 121 formulates its command only from CP2 information.

The invalidation command is sent from the SC to the respective BCE register 65 and trigger 66 in FIGURE 5. When set, the invalidation request trigger 66 activates the output from register 65 to the cache castout and invalidate controls 64, in order to specify a search of the BCE0 SIC directory 62 for the address having the class and set indicated in register 65.

The output of invalidation request trigger 66 simultaneously provides a signal to BCE priority circuit 28 to request the search of the SIC directory, since it is possible for simultaneous search requests to exist from other sources. When priority is signalled on line 29, controls 64 cause the search to immediately occur. When the search finds the required entry in the SIC directory, the signal from trigger 66 causes the valid bit in the found entry to be set to its invalid state to invalidate the corresponding line in SIC 63.

FIGURE 8 illustrates in detail the internals of the cache castout and invalidate control 64 in FIGURE 5. The format of the input signals to FIGURE 8 is shown in FIGURE 2C for the SC invalidate command. Invalidate control 64 simultaneously provides signals to the BCE directory to locate and invalidate the conflicting entry and provides signals to the command/data register 65 to formulate an invalidate command therein (shown in FIGURE 2D) for transmission to the SC to invalidate the corresponding entry in the associated copy directory, after the entry is invalidated in the associated BCE directory. Thus, in FIGURE 5, the request trigger 66 is set by an invalidation signal on line 71 (or by a castout signal (not shown) on a line 71A, because this invention is not concerned with the cast-out function. The output of invalidation trigger 66 is provided to FIGURE 8 to condition each of the AND gates 181, 182, 183 and 184. AND gates 181 and 182 pass the class and set address of the required entry to register 55 and to BCE0 directory 62 on lines 191 and 192. Gate 183 outputs any castout signal to directory 62 when in the SC invalidate command, there is no CL bit set but the castout bit is set. AND gate 184 provides the invalidating signal to directory 62 on line 193 when activated by the set CL bit signal on line 174 and the invalidate signal on line 171, but with the castout bit signal off on line 175. The output of AND gate 184 also controls the signal on line 194 which indicates a non-main-store request and OP code signals on the eight lines 195 when generator 190 is activated by the output of AND gate 184 to control the SC operation for invalidating the corresponding entry in the associated copy directory. A timing trigger 185 is also actuated by the output of AND gate 184 to provide a signal on line 186 that resets invalidation register 65 and invalidation trigger 66.

Furthermore in FIGURE 6B, the set output of the conflict triggers 124 or 136, or a remote SC conflict signal, passes through an OR circuit 138 to set a cancellation trigger 108 which then

provides a cancel request for the main storage command in the MS queue previously issued as a result of the clear line command. Also a release signal is provided on line 68 from the SC to the SAC trigger 81 in FIGURE 5 via an AND gate 78 conditioned by the CL trigger 58. At the same time, status register 95 in FIGURE 5 is set with a code indicating the conditions for the completion of the command. Lines 74 transmit the status information to a BCE status register and decoder 76 in FIGURE 5 to inform the BCE of current SC conditions, e.g. that the requested invalidation has been performed.

The release controls 93 are actuated by a signal from an OR circuit 140 in FIGURE 6A which receives a line 139A from AND gate 139 in FIGURE 6B. A signal is provided from OR circuit 138 and when no storage protect violation signal is provided from inverter 97.

FIGURES 9A through 9E illustrate the sequence of operations that occur with the hardware illustrated in FIGURES 4, 5, 6A and 6B. In FIGURES 9, IE(I) represents the BCE with the CPU issuing the CL command. BCE(R) represents a remote BCE which finds a conflict in its cache directory, but BCE(R) did not issue the CL command having the conflict. In FIGURES 9A through 9E the numerals in circles represent the time sequence of operations, in which the operation is indicated by the word(s) above a horizontal line and between adjacent vertical lines, and the statement below the horizontal line represents the transfer or location of the indicated operation.

FIGURE 9B represents the basic sequence of operations which occur from the initial issuance and from each reissuance of the CL command until the synchronized searches find no conflict in all SC

copy directories in the MP. FIGURE 9C represents the sequence of operations in the issuing IE, BCE and SC if an MP cache conflict is indicated by the output of OR circuit 138, in which case resend trigger 101 transmits a resend signal back to the issuing IE unit, i.e. IE(I), which causes IE(I) to reissue the same CL command to BCE(I), instead of the next command that the IE(I) would have issued. When the resend command is reissued, the basic operations represented in FIGURE 9B are repeated.

In FIGURE 9B, after the resend signal is provided, the BCE(I) may not immediately accept the CL command being reissued while the BCE(I) is performing the invalidate line (IL) operations represented in FIGURE 9C; and after their completion, BCE(I) permits the transfer of the reissued CL command into its command register 55.

FIGURE 9A represents the final sequence of operations whenever the simultaneous search in all copy directories during time 5 in FIGURE 9B finds that no MP conflict exists, in which case accept trigger 104 is set which then causes IE(I) to transfer the double word of pad data in its data register 17 to BCE(I) register 55 in FIGURE 5, which is then transmitted to the SC(I) PIR 87 in FIGURE 6A and then to main storage via gate 89 and storage input register (SIR) 90 on succeeding cycles, as represented in FIGURE 9A.

Most CL command requests are likely not to find any conflict, in which case only the sequences in FIGURES 9B and 9A occur. Whenever a conflict is found in any copy directory, the sequences in FIGURES 9C and 9D also occur, and the sequence in FIGURE 9E occurs if any remote SC is busy, which causes the resend command to issue in SC(I).

The other aspects of the sequence diagrams in FIGURES 9A through 9E are considered self-explanatory in view of the prior explanation of the embodiment.

FIGURE 7 illustrates the manner in which a BSM controller (BSC) may accumulate a line of data from an SC a double word (DW) at a time alternately through an odd DW register 171 and an even DW register 172, until a line of 16 doublewords is accumulated in a BSM line register 151. Then register 151 stores the line as a unit into a BSM board.

The writing of one line into the BSM completes the execution of one CL command. The execution of the SVP CL instruction or a user instruction such as the MVCL results in the execution of plural CL commands as determined by the IE in the manner previously explained.

FIGURE 10 shows another arrangement of the BSC for handing a clear line operation which enables faster access to MS than the BSC arrangement in FIGURE 7.

FIGURE 10 provides a parallel replication of the pad DW in data register 162 in one machine cycle instead of the serial replication requiring 16 machine cycles done in the BSC in FIGURE 7. In FIGURE 10, when the CL trigger 163 in the BSC is set by a CL command in the BSC command register 161, the pad DW in register 162 is transferred in a fanout manner into all sixteen DW locations in BSM line register 151 during the one machine cycle of enablement of gates 180-0 through 180-15 by CL trigger 163.

In a UP where one copy directory exists, or no copy directory exists, the invalidation in the processor cache directory can occur when the CL command is issued without the SC(I) and BCE(I) in-

validate line (IL) commands being used as previously discussed (which are required in a MP). Thus, in a UP, the inhibit gate 59 may be connected to the miss output of the directory 62 to inhibit the formation of any line fetch request from the SIC to MS but can allow a search of the processor cache directory for locating the existence of the line to be invalidated (but not for determining the absence of the line for a fetch request).

CLAIMS

1. A data processing system including a plurality of central processing units (CP0 to CP3), each having a store in cache (13), a cache directory (62) and store in cache control means (57, 59, 64, to 66), and including system control means (SC0, SC1) coupled between the processing units and a shared main storage system (MS), characterised in that the system control means includes a plurality of copy directories (116, 117), each being a copy of an associated one of the cache directories, first means (91, 114, 131) for simultaneously searching all of the copy directories for an address contained in a clear storage command issued by a processing unit, second means (120, 121, 123, 135) coupled to the copy directories for generating invalidation signals for each copy directory containing the searched address and coupled to the respective cache directories for invalidating the corresponding address entries therein and third means (142, 151 to 154) coupled to the copy directories to provide a resend signal to the processing unit issuing said command when any of the copy directories contain the searched address or an accept signal thereto when none contain said address, and in that each said store in cache control means includes means (64, 65) responsive to said invalidation signals to invalidate the cache directory address defined by said command and means (64) responsive to said invalidation to invalidate that address in the corresponding copy directory, said processor issuing said command being responsive to a resend signal to re-issue the command to initiate a copy directory search or to an accept signal to clear the main storage area defined by the command.

2. A system according to claim 1, further characterised in that the main storage area defined by a command is equal to the area invalidated in the directories.

3. A system according to claim 1 or claim 2 further characterised by search inhibit means (59) in each store in cache control means to inhibit a search of the cache directory directly in response to the issuance of a clear storage command by the associated processing unit.

4. A system according to any of the previous claims further characterised in that said resend signal is operative to prevent the receiving processing unit from issuing any other command.

5. A system as claimed in any of the previous claims further characterised in that each cache control means includes priority means (28) to determine the next command or invalidation request to access the cache directory.

FIG. 1

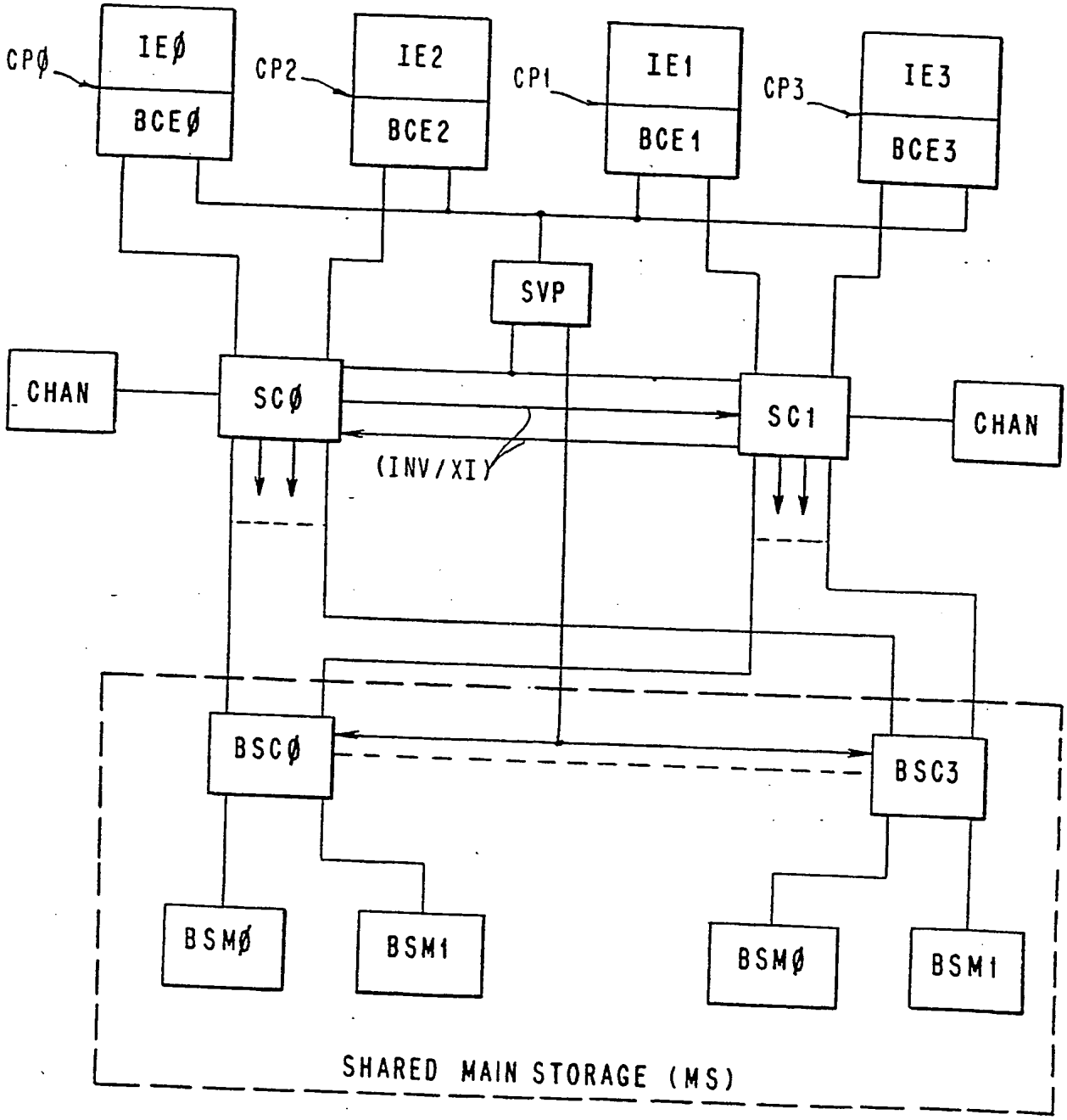

CPØ
IEØ
BCEØ
CP2
IE2
BCE2
CP1
IE1
BCE1
CP3
IE3
BCE3
SVP
CHAN
SCØ
SC1
CHAN
(INV/XI)
BSCØ
BSC3
BSMØ
BSM1
BSMØ
BSM1
SHARED MAIN STORAGE (MS)

FIG. 2A IE CLEARLINE (CL) CMD TO BCE

PSW STORAGE KEY	LOGICAL ADDRESS	SPECIAL ACCESS BIT	CLEARLINE OP CODE
(4 BIT)	(32 BITS)	(1 BIT)	(8 BITS)

FIG. 2B BCE CLEARLINE CMD TO SC

PSW STORAGE KEY	ABSOLUTE ADDRESS	#DW (0001)	CL OP CODE	CPU ID
(4 BITS)	(27 BITS)	(4 BITS)	(8 BITS)	(2 BITS)

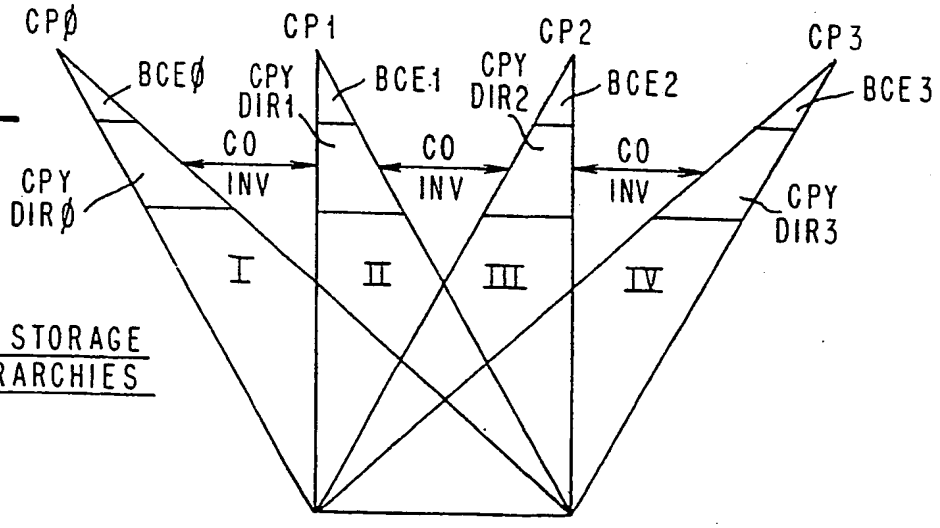
FIG. 2C SC INV / CO CMD TO BCE

DIRECTORY CLASS ADDRESS	DIRECTORY SET ADDRESS	CLEARLINE BIT	CASTOUT BIT	INVALIDATE BIT
(8 BITS)	(4 BITS	(1 BIT)	(1 BIT)	(1 BIT)

FIG. 2D BCE INVALIDATE SC DIRECTORY ENTRY CMD

DIRECTORY CLASS ADDRESS	DIRECTORY SET ADDRESS	NON MS REQ	OP CODE FOR INVALIDATING SC DIR ENTRY	CP ID
(8 BITS)	(4 BITS)	(1 BIT)	(8 BITS)	(2 BITS)

FIG. 3



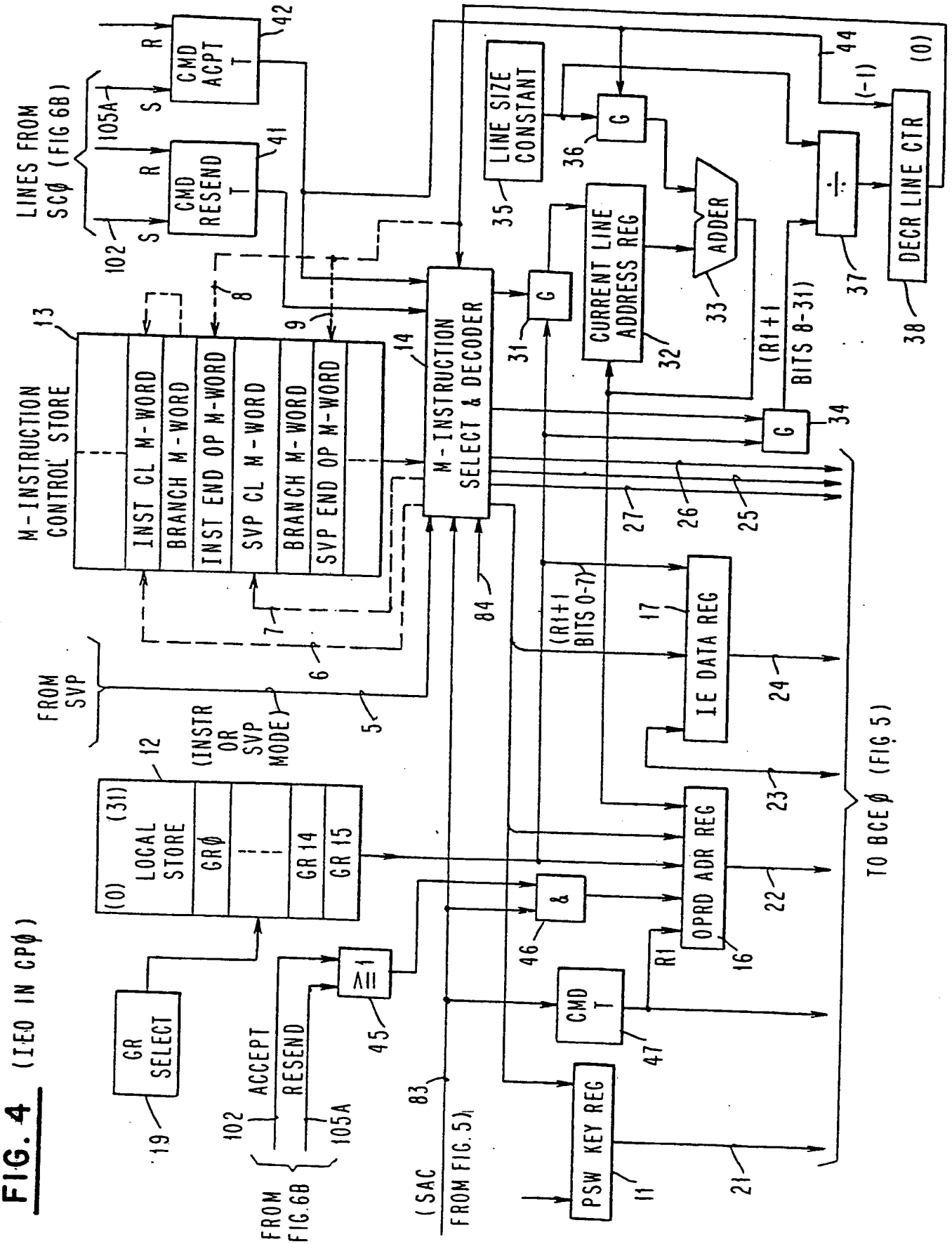
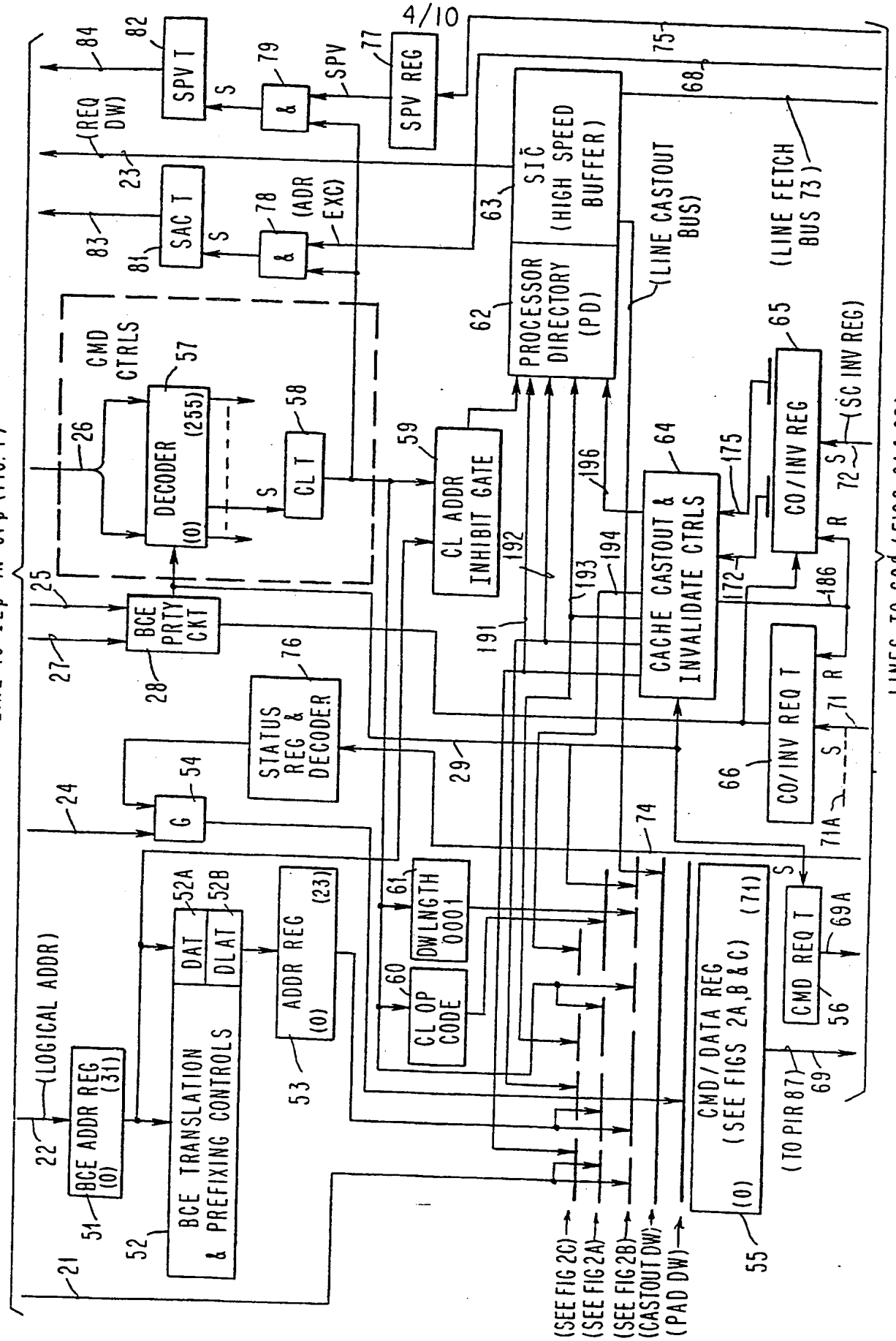
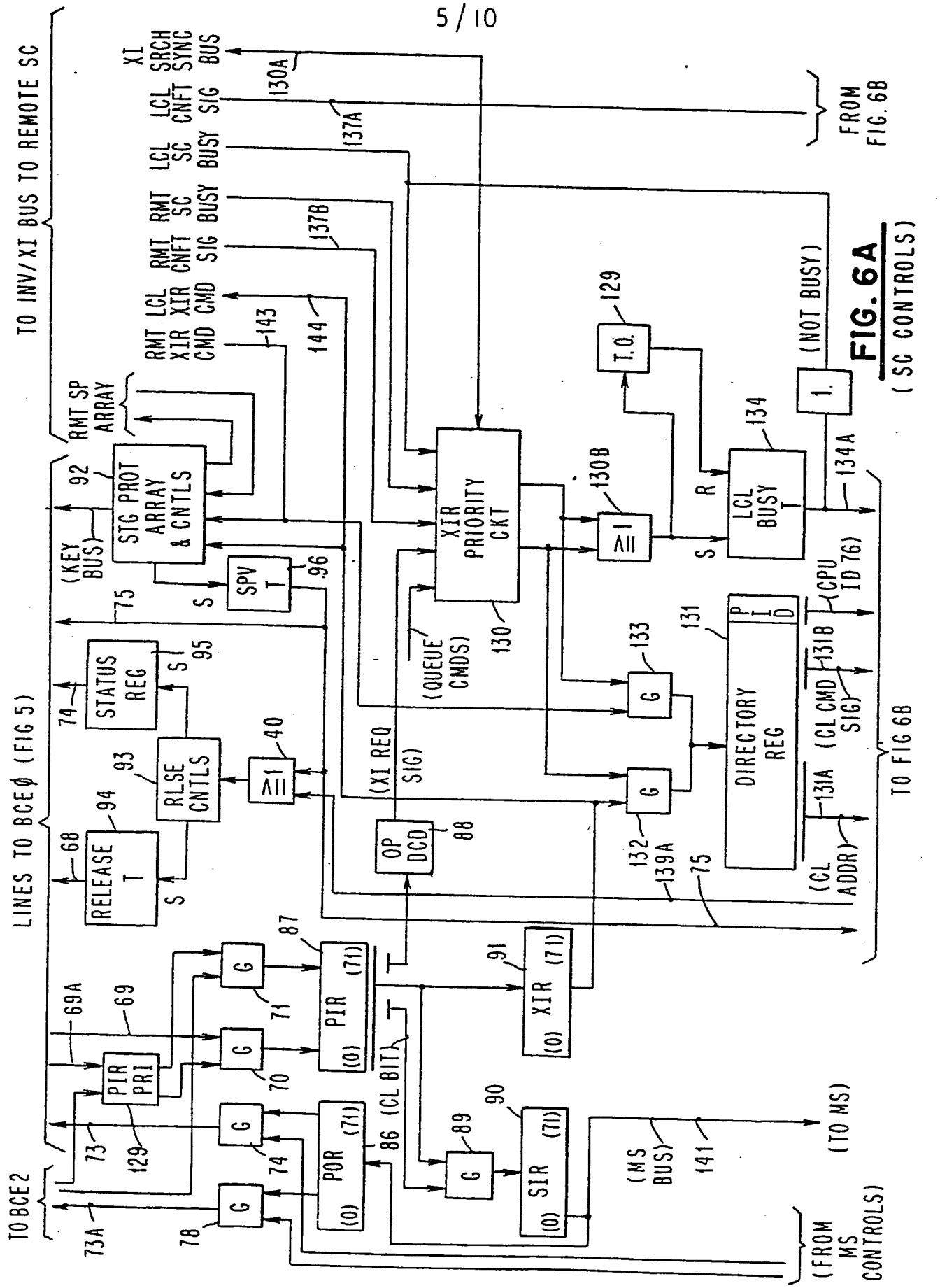


FIG. 4 (IEO IN CPØ)

FIG. 5 (BCE)


LINE TO IEØ IN CPØ (FIG. 4)
(LOGICAL ADDR)
BCE ADDR REG (31)
BCE TRANSLATION & PREFIXING CONTROLS
DAT
DLAT
ADDR REG (23)
STATUS REG & DECODER
BCE PRTY CKT
CMD CTRLS
DECODER (255)
CL T
SAC T
SPV T
SPV REG
CL ADDR INHIBIT GATE
CL OP CODE
DWLNGTH 0001
PROCESSOR DIRECTORY (PD)
SIC (HIGH SPEED BUFFER)
CACHE CASTOUT & INVALIDATE CTRLS
CO/INV REG
CO/INV REQ T
CMD/DATA REG (SEE FIGS 2A,B & C)
CMD REQ T
(LINE CASTOUT BUS)
(LINE FETCH BUS 73)
(SC INV REG)
(REQ DW)
(ADR EXC)
(TO PIR 87)
(SEE FIG 2C)
(SEE FIG 2A)
(SEE FIG 2B)
(CASTOUT DW)
(PAD DW)
LINES TO SCØ (FIGS 6A & 6B)


TO INV/XI BUS TO REMOTE SC
XI SRCH SYNC BUS
LCL CNFT SIG
LCL SC BUSY
RMT SC BUSY
RMT CNFT SIG
RMT LCL XIR CMD
RMT SP ARRAY
STG PROT ARRAY & CNTLS
(KEY BUS)
SPV T
XIR PRIORITY CKT
(QUEUE CMDS)
T.O.
LCL BUSY T
(NOT BUSY)
STATUS REG
RLSE CNTLS
RELEASE T
OP DCD
(XI REQ SIG)
DIRECTORY REG
CPU ID 76)
(CL CMD SIG)
(CL ADDR)
PIR PRI
PIR (71)
XIR (71)
POR (71)
SIR (71)
(CL BIT)
(MS BUS)
TO BCE2
LINES TO BCEØ (FIG 5)
(FROM MS CONTROLS)
(TO MS)
FROM FIG. 6B
TO FIG 6B
FIG. 6A
(SC CONTROLS)

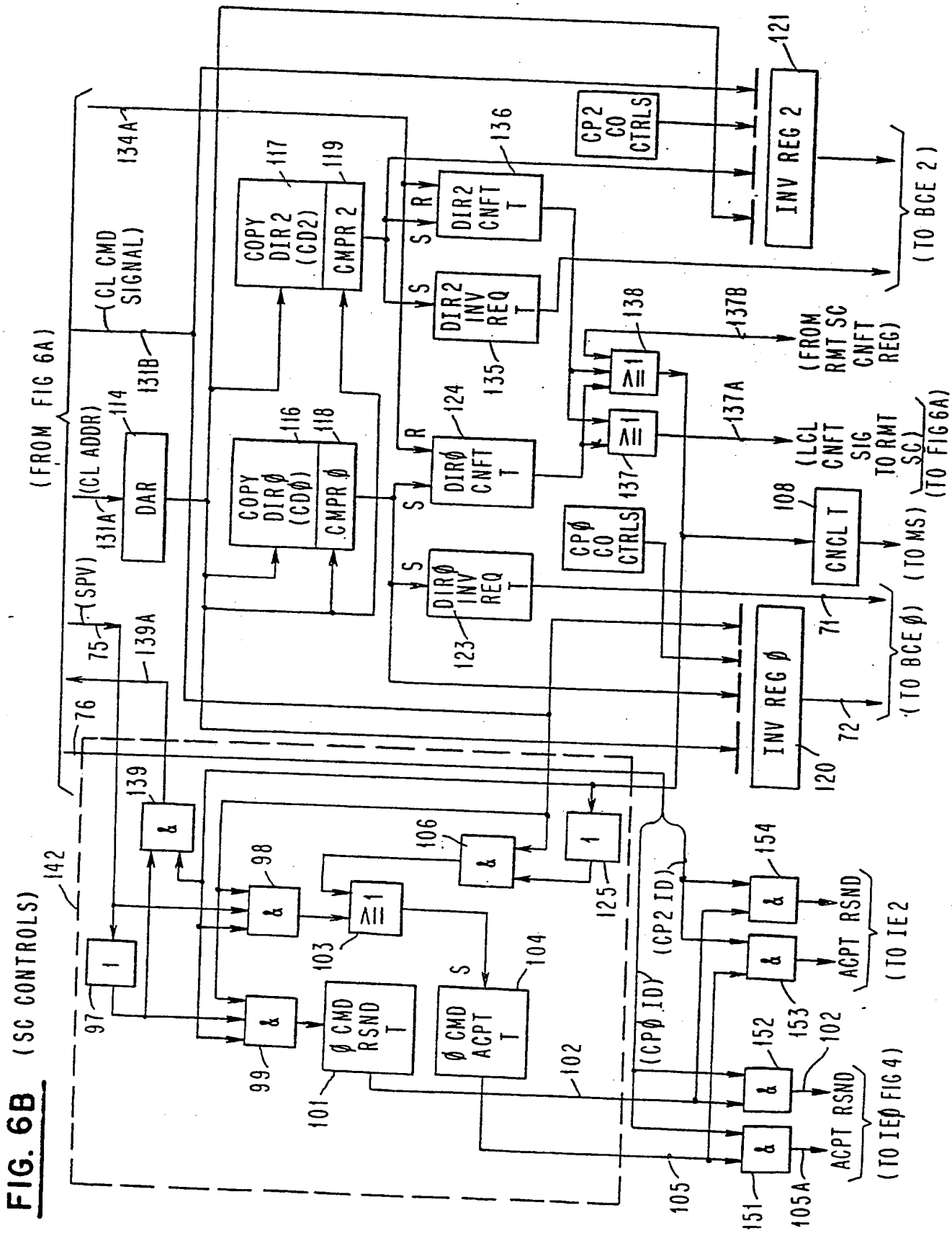

(FROM FIG 6A)
(SC CONTROLS)
(CL CMD SIGNAL)
(CL ADDR)
(SPV)
DAR
COPY DIR2 (CD2)
CMPR 2
COPY DIRØ (CDØ)
CMPR Ø
DIR2 CNFT T
DIR2 INV REQ T
DIRØ CNFT T
DIRØ INV REQ T
CP2 CO CTRLS
CPØ CO CTRLS
INV REG 2
INV REG Ø
CNCL T
(TO BCE 2)
(TO BCE Ø)
(TO MS)
(FROM RMT SC CNFT REG)
(LCL CNFT SIG TO RMT SC)
(TO FIG 6A)
Ø CMD RSND T
Ø CMD ACPT T
(CP2 ID)
(CPØ ID)
ACPT RSND
(TO IE 2)
(TO IEØ FIG 4)

FIG. 6B

FIG. 7

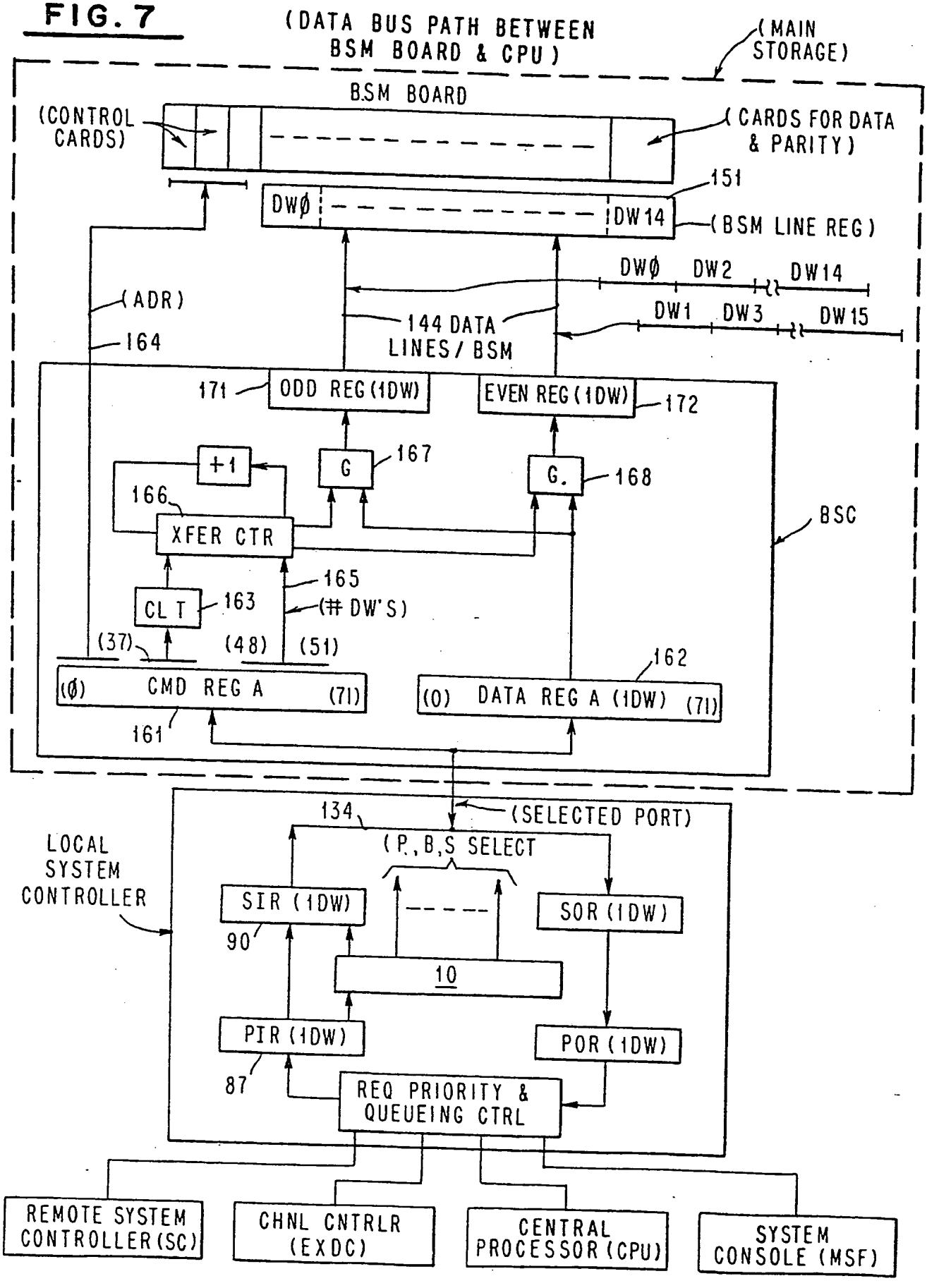

(DATA BUS PATH BETWEEN BSM BOARD & CPU)
(MAIN STORAGE)
BSM BOARD
(CONTROL CARDS)
(CARDS FOR DATA & PARITY)
151
DWØ
DW14
(BSM LINE REG)
DWØ DW2 DW14
DW1 DW3 DW15
(ADR)
164
144 DATA LINES/BSM
171
ODD REG (1DW)
EVEN REG (1DW)
172
+1
G
167
G.
168
166
XFER CTR
BSC
165
CLT
163
(# DW'S)
(37)
(48)
(51)
162
(Ø) CMD REG A (71)
(0) DATA REG A (1DW) (71)
161
134
(SELECTED PORT)
(P,B,S SELECT
LOCAL SYSTEM CONTROLLER
SIR (1DW)
SOR (1DW)
90
10
PIR (1DW)
POR (1DW)
87
REQ PRIORITY & QUEUEING CTRL
REMOTE SYSTEM CONTROLLER (SC)
CHNL CNTRLR (EXDC)
CENTRAL PROCESSOR (CPU)
SYSTEM CONSOLE (MSF)

FIG. 8

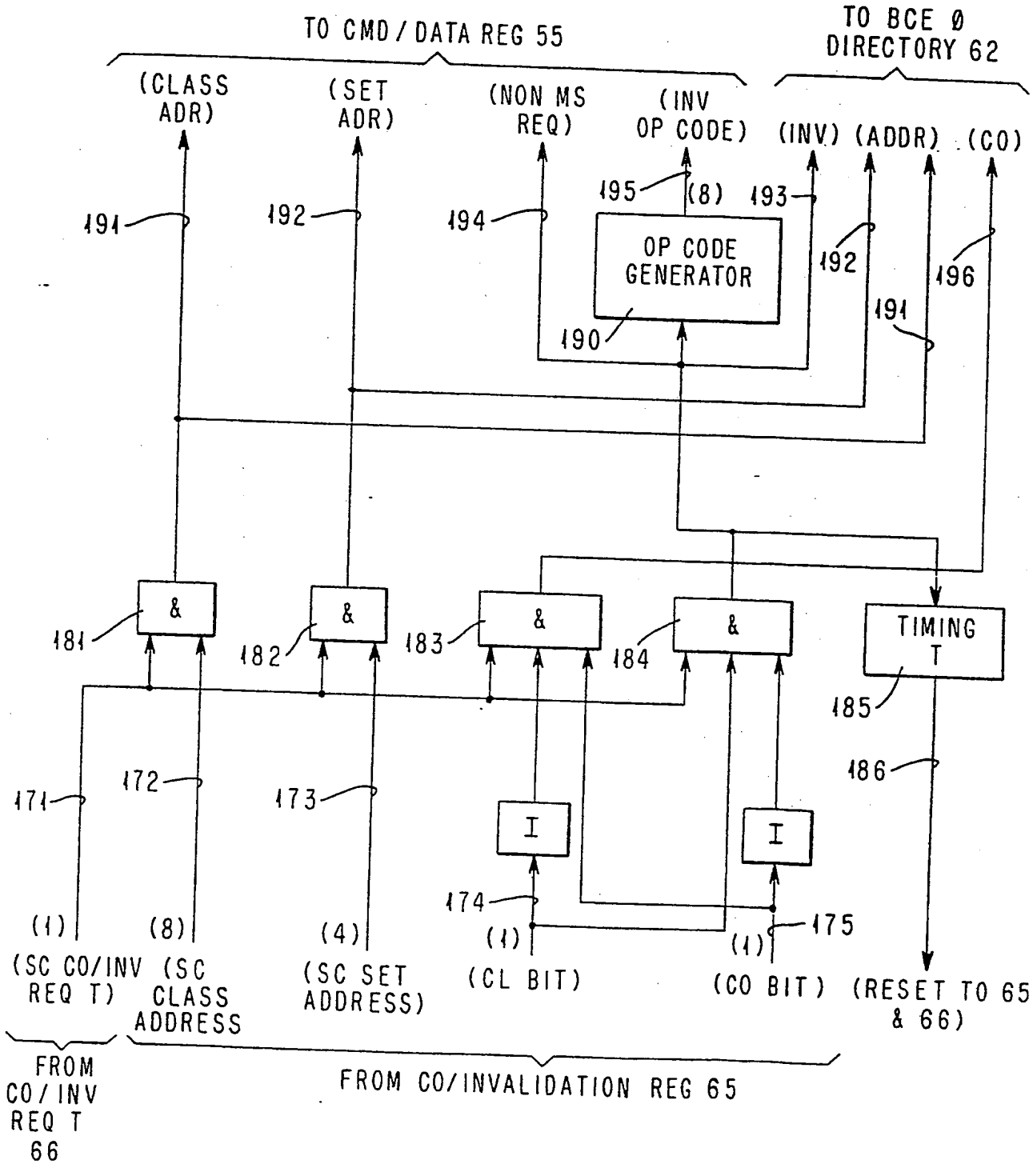
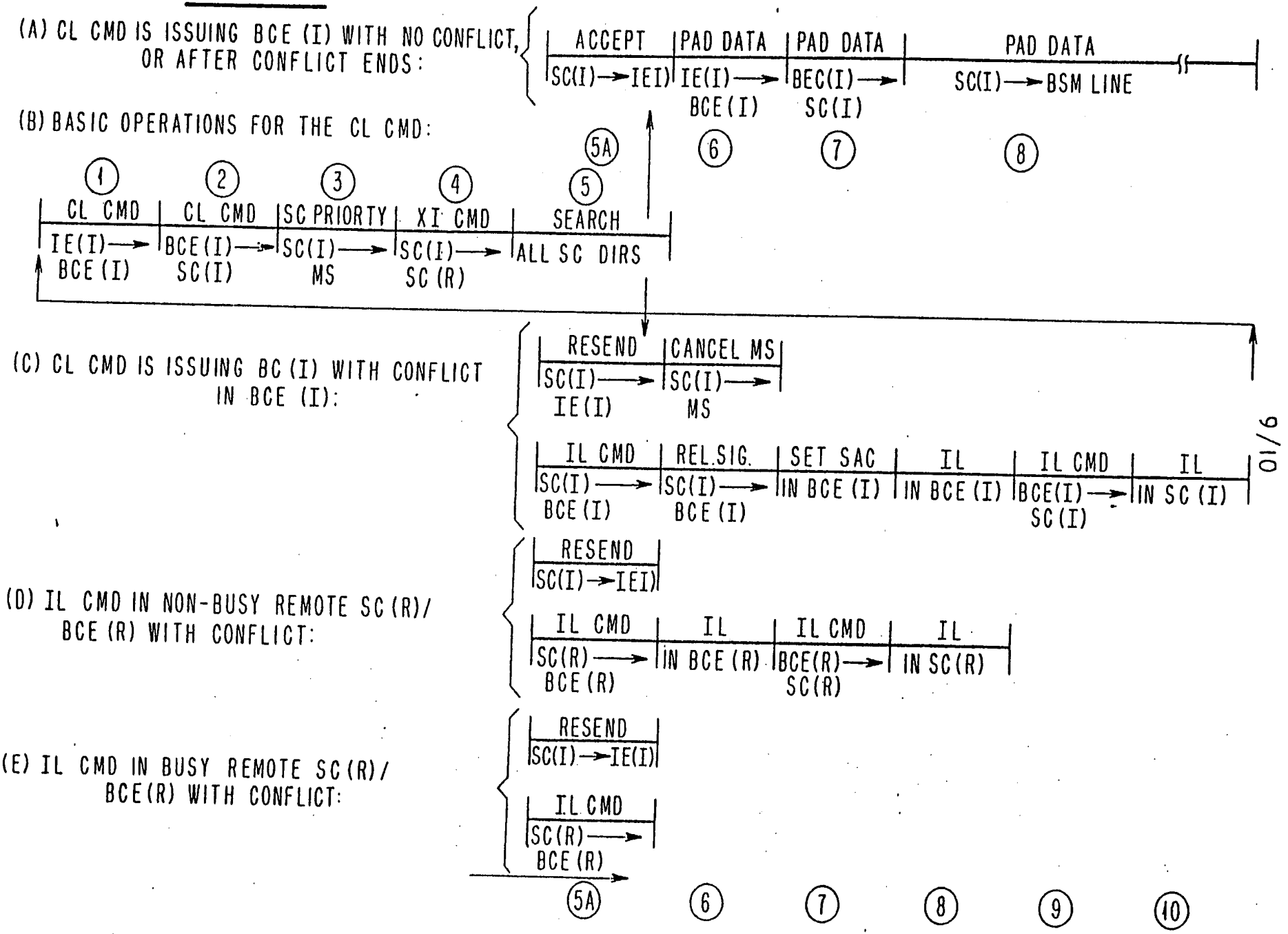

(CACHE CASTOUT & INVALIDATE CONTROL 64)
TO CMD / DATA REG 55
TO BCE Ø DIRECTORY 62
(CLASS ADR)
(SET ADR)
(NON MS REQ)
(INV OP CODE)
(INV)
(ADDR)
(CO)
191
192
194
195
(8)
193
OP CODE GENERATOR
190
196
&
181
182
183
184
TIMING T
185
186
171
172
173
I
174
175
(1)
(8)
(4)
(1)
(1)
(SC CO/INV REQ T)
(SC CLASS ADDRESS)
(SC SET ADDRESS)
(CL BIT)
(CO BIT)
(RESET TO 65 & 66)
FROM CO/INV REQ T 66
FROM CO/INVALIDATION REG 65

FIG. 9



01/6

0049387

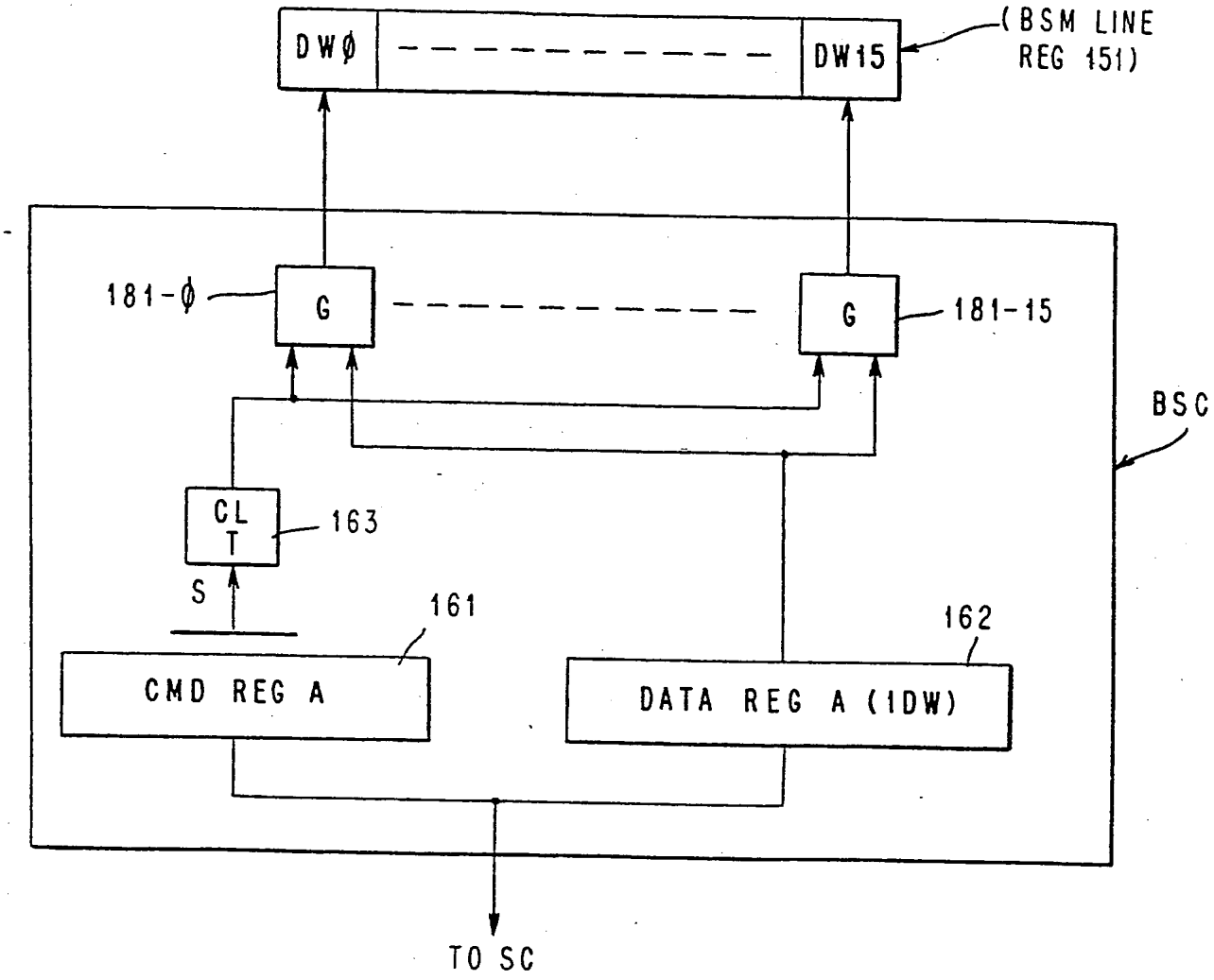



FIG. 10 BSM CONTROLLER (BSC)